Open Pedigree Standard Specification

Version 5, 14th July 2003

I General Information

I.1 Revision History

Version 1, 21st November 2001: Initial Draft.
Version 2, 3rd December 2001, First Revision:

- renamed SourceIn, DateIn, SourceUp and DateUp to be “InputSource”, “InputDate, “UpdateSource”, and “UpdateDate” and also moved these fields to be fields common to any record type. (not specific to animals).
  - subfield separator changed to be &#027; not

  - added section about multiple field values and added columns to the field tables to indicate which fields are anticipated may have multiple values for a single record
  - renamed the binary field type to be ‘binary’ rather than ‘hex’
  - defined a mechanism by which binary data fields may be encoded in ways other than hex; this is really intended as a placeholder for defining the attribute that is to be used in future if non-hex encoded binary data is added to the spec; this can also affect the encoding of other binary-style fields such as JPEG images. Note: this does not give us any more work in implementing v1 of the spec!
  - put back the overall file structure to be what Ron initially proposed
  - changed the currency field type as per Ron’s suggestion. Now much simpler.
  - addition of maxsz attribute for text fields
  - addition of encryption attribute

Version 3, 8th March 2002, Second Revision:

- File Naming Convention: MIME type added
- <table> tag within <data> renamed as <t>
- ‘id’ attribute renamed as ‘tid’ (<t> and <table> tags) or ‘fid’ (<f> and <field> tags)
- added text to state which attributes are required and which are optional

- ‘alfirin’ source id defined
- character set now defined as iso-8859-15
- datetimes: now iso 8601 compliant using UTC time
- explicit note that whitespace is ignored in binary data
- MIME types now used to identify graphics file formats
- Support for field data not being inline, but specified indirectly by use of the XLink recommendation
- ‘type’ attribute may optionally be used in <f> tag, e.g. when used in image fields with inlined image data.

Version 4, 10th July 2003:

- Some additional animal types added to ANIMALID, section 1.4
- Section 1.4, OPSVERSION incremented to 2 due to incompatible changes from earlier spec
- Section I.5.A, Multi-Line Data - changed the handling of raw newline characters, now they are treated as whitespace.
- Section I.5.B removed – Sub-Field Separators
- New Section I.12 added – Personal Names and Addresses. Describes the handling of name and address fields in split and combined format fields.
- New Section I.13 added – Field Grouping.
- Section II.2 and II.2 – new fields added resulting from the addition split name/addresses
- New Section I.2.A – Terminology
- Section I.6 – Non-Inline Data – note added making it clear that full support for inline binary image data is optional.
- Sections II.1, II.2, II.3 – Standard Fields – All fields renumbered, new fields introduced due to new split/combined names+addresses, and removal of sub-field separators.

Version 5, 15th July 2003:

- Addition of Copyright Statement
- Section 1.7 – private tags and attributes now start with underscore ‘_’ not ‘x’
- Section 1.9A – removal of maxsz attribute and references to sub-field separators
- Section 1.9C – underscore (_) now used instead of Z to separate date from time
- Section 1.9 – sections relating to image files now contain explicit statement saying that OPSX compliant apps are not required to implement any specific image format, though they must be able to parse them within OPSX files.
- Section 1.4 – addition of SHORTCODE
- Section II – removal of ‘short’ versions of some fields, replaced with permissable use of ‘short’ attribute on the equivalent ‘full’ fields

I.2 Overview

This document is the definition of the Open Pedigree Standard (OPS) – an open format for the interchange of animal pedigree information. The standard is based on XML. Anybody is free to use this specification for whatever purpose they like.

It is recommended that any product, such as a pedigree software package, should both read and write files of this format, so as to maximize the benefit to users of the standard.

This standard was developed by a group of pedigree software suppliers to answer the need for a pedigree interchange format independent of any one supplier and that is sufficiently rich to enable compliant programs to exchange data easily.

The standard is intended to be neutral of any specific operating system platform, e.g Windows.

I.2.A Terminology

This section lists some terms used in this document:

A ***writer*** is any application that generates an OPSX file.

A ***reader*** is any application that processes OPSX files.

I.3 File Naming Convention

OPSX files are XML (eXtensible Markup Language) files and as such will have the extension .xml.

OPSX files have MIME type "application/x-pedigree+xml"

I.4 File Structure

An OPSX file is an XML file with the following structure:

```
<?xml version="1.0"?>
<opsg version='OPSVERSION' source='SOURCEID' sourcever='string' animal='ANIMALID'
description='DESCRIPTION'>
<definition>
  <table name='TABLENAME' tid='TABLEID'>
    <field name='FIELDNAME' fid='FIELDID' type='TYPEID'/>
    .
    . repeat for other field definitions for this table
    .
  </table>
.. definitions for other data structures (other than tables) could be added here
</definition>
<data>
  <t name='TABLENAME' tid='TABLEID'>
     <record>
       <f fid='FIELDID' type='TYPEID' short='SHORTCODE'>FIELDDATA</f>
       .
       . repeat for other fields within this record
       .
     </record>
     .
     .  repeat for other records in this table
     .
  </t>
..
.. repeat for other tables
..
  data for other data structures (other than tables) could be added here
</data>
.
</opsg>
```

where:

OPSVERSION	This is a required attribute. Represents the OPS standard version to which the file complies. Initially this is fixed at 2. It will be incremented as newer versions of the standard are defined.

SOURCEID	This is a required attribute. Is a string defining the name of the generating program. Anyone may apply to the OPSG to have a SOURCEID allocated to them. A SOURCEID consists of alphabetic and numeric characters only. Standard SOURCEID's will never start with the character '_', which is reserved for use by applications which have not applied to OPSG for their own SOURCEID to be allocated. (i.e _ prefaces a private use of the standard) The following standard sourceid's are defined: In alphabetical order: source='breedersassistant' source='breedmate' source='k9ped' source='pedigrees2000' source='alfirin'
SOURCEVER	Is an optional attribute. The format of the attribute value is undefined. The purpose of the attribute is to identify the version of the generating (source) program. When a program processes an OPSX file, the value of the SOURCEVER attribute should only be programmatically interpreted when the program processing the file is the same product as generated the file. E.g it might use this attribute to identify which private field types to expect etc.
ANIMALID	This is a required attribute. It is a string defining the kind of animal. Anyone may apply to the OPSG to have an ANIMALID allocated. An ANIMALID consists of alphabetic and numeric characters only. Standard ANIMALID's will never start with the character '_', which is reserved for private use by applications. The following standard ANIMALID's are defined: animal='bird' animal='cat' animal='cavy' animal='dog' animal='hamster' animal='horse' animal='rabbit' animal='rat' animal='sheep' animal='goat' animal='cow' animal='pig' animal='undefined'

DESCRIPTION	This is an optional attribute. It may be used to describe the data within the file. Typically this might be presented to the user of a program that is importing the file. E.g. ‘Johns latest cat data’ or 'That data you asked for' or 'Persians, 2000-2001' etc
TABLENAME	This is an optional attribute. Is a placeholder for a description of the purpose of the table. It serves no other purpose except to assist a human in browsing through an OPSX file, or to present in a user interface by a program that is processing the file. OPS compliant programs use the TABLEID to identify the purpose of a table.
TABLEID	This is a required attribute. Is an integer number that identifies the purpose of the table. This document defines standard values for some TABLEID’s. Such standard values are always positive integers, and the use of positive integers for TABLEID’s is reserved by the standard. An OPS compliant program may include private tables in an OPSX file by assigning them negative values.
FIELDNAME & FIELDID	Work on similar lines to TABLENAME and TABLEID except that they work at the field level. Again, positive FIELDID’s are reserved for definition by this standard. The only exception to this is within a table that itself is private (ie negative TABLEID). FIELDID is a required attribute. FIELDNAME is an optional attribute.l
TYPEID	This is a required attribute when used in a <field> tag (i.e., within a field definition). It is optional when used in a <f> tag (i.e., field data within a record). Indicates the ‘content’ of the data for a field. Standard field data types are defined in I.8. A program may use non-standard TYPEID’s so long as they start with the letter ‘_’. It should be noted that for many of the standard fields, the TYPEID is defined implicitly in this document. And therefore the type= attribute need not be specified within every OPSX file. However, they can (and perhaps should) be included within field definitions for completeness.
FIELDDATA	Defines the field data for a field.
SHORTCODE	May be set to ‘1’ to indicate a short form of a field. Used with certain fields that may be presented in both ‘normal’ (long) form and a short or abbreviated form.

Fields that support this attribute are indicated in section II.

I.5 Character Set

An OPSX file is encoded using iso 8859-15

Notes
*1. To include the tab character as actual field data, the 	 HTML character entity must be used.*
2. OPS does not presently support far eastern languages.

I.5.A Multi-Line Data

New lines within the data of a field are treated as if they were a single space. Thus:

```
<f fid='1'>Hello There</f>
```

has exactly the same meaning as

```
<f fid='1'>Hello
There</f>
```

When field data represents true multi-line text, the HTML character entity for carriage return should be used whenever a 'real' new line is to be started within the data. i.e
 Thus:

```
<f fid='205'>1 High Street&#013;Mytown&#013;Anystate 12345&#013;US</f>
```

represents multi-line data for a 'combined' address like this:

```
1 High Street
Mytown
Anystate 12345
US
```

I.6 Non-Inline Data

Field data is normally supplied inline within a <f> tag. E.g.

```
<f fid='802' type='image/bmp'>8762348768…</f>
```

where the data in this case is a binary encoded.

However, it may be supplied remotely by way use of attributes conforming to the XLink standard. E.g.

```
<f fid='802' xlink:href=http://foo.bar.com/images/dog.jpg type='image/bmp' />
```

NOTE: It is optional whether a reader fully supports inline binary data for images. An application may opt to only support images given as filenames (i.e., with type='string'). However, all readers must be able to parse fields with inline binary data, even if they just ignore it.

I.7 Non-Standard Attributes and Non-Standard Tags

Any application of OPSX format is free to use its own non-standard (private) tags and attributes. All such tags and attributes must start with the letter '_'.

e.g. a non-standard attribute used within a field definition:

```
<field name='Name' fid='1' xcolour='FF0000'>
```

e.g. a non-standard tag:

```
<xmyprivatedata>blah blah blah</myprivatedata>
```

I.8 Multiple Field Values

Certain fields can have multiple values for a single record. E.g. with an animal record, you can have multiple breeders, or multiple pre-titles, or multiple registration numbers.

Such fields are permitted to be given multiple times within a single record, once for each data value. E.g.

```
<f fid='520'>Champion</f>
<f fid='520'>Premier</f>
```

The sections below indicate those fields where multiple values are anticipated.

Where multiple instances of the same field id are given within a single record, AND if it is meaningful for the values to be ordered in some kind of 'priority', this ordering is implied by the ordering of the fields within the record. E.g if there is a 'primary' and a 'secondary' registration number (that is less important) then one would expect the exporting program to write the field containing the primary registration number first.

Note also that grouping is often used in conjunction with multi-valued fields, so as to present multiple groups of associated fields. E.g. multiple registration information could be presented like this:

```
<g>
  <f fid='530'>CS987987</f>
  <f fid='531'>GCCF</f>
  <f fid='532'>20020830</f>
  <f fid='533'>UK</f>
</g>
<g>
  <f fid='530'>SBT 987987</f>
  <f fid='531'>TICA</f>
  <f fid='533'>US</f>
</g>
```

etc

I.9 Standard Field Types

I.9.A Strings

```
type='string'
```

A string that is otherwise uninterpreted. This is the default field type for all standard fields unless otherwise specified. Multi-line text can be included by way of the HTML character entity for carriage return (see I.5.A).

I.9.B Dates

```
type='date'
```

Dates are encoded as yyyymmdd e.g

```
20010329
```

I.9.C Date Times

```
type='datetime'
```

Datetimes are encoded as yyyymmdd_hhmmss e.g

```
20010329_235959
```

would be 23:59.58 on 29th March 2001. Datetimes should be given in Universal Coordinated Time (UTC) (GMT).

I.9.D Boolean or Logical Values

```
type='bool'
```

Boolean values encoded as:
'0': false
'1': true

I.9.E Integer Numbers

```
type='int'
```

An integer number. Negative numbers allowed.

I.9.F Floating Point Numbers

```
type='float'
```

An floating point number in printf '%g' format.

I.9.G Binary Data

```
type='binary' [format='hex']
```

Binary data. The 'format' attribute is an optional attribute that states the format in which the data is present. At present, only one format is defined, which is 'hex' in which the bytes are presented in hex code in the usual way. Irrespective of the value of the 'format' attribute, the actual data for the field will be encoded using only printable 7bit ASCII characters (codes 33 though 126), excluding all characters that have special meaning in XML (like '<'). Whitespace characters are ignored. If the 'format' attribute is not explicitly given, 'hex' format is assumed. E.g. with hex format:

```
'0134F7e6'
```

would represent the following four bytes of data:

```
0x01
0x34
0xF7
0xe6'
```

Upper/lower case in hex encoding is immaterial.

Private formats may be used by an application by using a value for the format attribute that starts with '_', e.g:

```
<field fid=… type='binary' format='_myformat'>
```

Note:
The purpose of the 'format' attribute is to define the means by which additional formats for binary data will be presented within an OPSX file. Upon reading an OPSX file, if an importing program finds there is a format attribute that it does not understand, it can at least handle the situation gracefully.

I.9.H JPEG Image Files

```
type='image/jpeg' [format='hex']
```

A JPEG image. The format attribute is optional, but if present works in exactly the same way as for binary data (see I.9.G). When the format attribute is not defined for a field of this type, the field data is encoded in hex.
The standard allows for the image data to be supplied inline, as above, but it can also be supplied indirectly by using xlink:href attributes (see above).
There is no requirement that OPSX compliant programs implement JPEG format images, but they must support the parsing of such images within an OPSX file.

I.9.I Windows Bitmap Image Files

```
type='image/bmp' [format='hex']
```

A BMP image, encoded in hex. The format attribute is optional, but if present works in exactly the same way as for binary data (see I.9.G). When the format attribute is not defined for a field of this type, the field data is encoded in hex.
The standard allows for the image data to be supplied inline, as above, but it can also be supplied indirectly by using xlink:href attributes (see above).
There is no requirement that OPSX compliant programs implement Windows bitmap format images, but they must support the parsing of such images within an OPSX file.

I.9.J GIF Format

```
type='image/gif' [format='hex']
```

A GIF image, encoded in hex. The format attribute is optional, but if present works in exactly the same way as for binary data (see I.9.G). When the format attribute is not defined for a field of this type, the field data is encoded in hex.
The standard allows for the image data to be supplied inline, as above, but it can also be supplied indirectly by using xlink:href attributes (see above).
There is no requirement that OPSX compliant programs implement GIF format images, but they must support the parsing of such images within an OPSX file.

I.9.K Windows Metafile Format

```
type='application/x-msmetafile' [format='hex']
```

A WMF image, encoded in hex. The format attribute is optional, but if present works in exactly the same way as for binary data (see I.9.G). When the format attribute is not defined for a field of this type, the field data is encoded in hex.
The standard allows for the image data to be supplied inline, as above, but it can also be supplied indirectly by using xlink:href attributes (see above).
There is no requirement that OPSX compliant programs implement Windows metafile format images, but they must support the parsing of such images within an OPSX file.

I.9.L Windows Enhanced Metafile Format

```
type='application/x-msenhancedmetafile' [format='hex']
```

A EMF image, encoded in hex. The format attribute is optional, but if present works in exactly the same way as for binary data (see I.9.G). When the format attribute is not defined for a field of this type, the field data is encoded in hex.
The standard allows for the image data to be supplied inline, as above, but it can also be supplied indirectly by using xlink:href attributes (see above).

There is no requirement that OPSX compliant programs implement Windows extended metafile format images, but they must support the parsing of such images within an OPSX file.

NOTE: there doesn't appear to be a formally defined MIME type for MS enhanced metafile. If anyone knows of any 'official' MIME type for this, please let us know!

I.9.M Amounts of Money

```
type='currency' [format='…']
```

Used for amounts of money. In a field definition for a field of type 'currency', the format attribute is a string specifying the layout. It may contain the placeholders %a and %b which represent the integral and fractional parts respectively. The integral part may be negative. If the format attribute is not given it is assumed to be '$%a.%b'.

E.g 1

If:

```
format='$%a.%b'
```

Then examples of valid amounts are:

```
$1.23
$100000.00
$-100.00
```

Examples of invalid amounts with this format include:

```
$1 (no fractional part)
1.23 (no $)
```

E.g 2

If:

```
format='%a,%b DM'
```

Then examples of valid amounts are:

```
1,23 DM
100000,00 DM
-100,00 DM
```

Notes

1. Spaces may be included within currency values but are ignored.

2. The purpose of the 'format' attribute is simply to define a way to present the values of amounts of money in a way that can be programmatically understood by an OPS compliant program if it so wishes. There is no requirement that a program actually uses these features. E.g. a product might use simple text fields to store currency values.

I.10 Indeterminate Field Values

If the value of a standard field is not known for a given record, this is represented in an OPSX file by that field being entirely missing within the record. Note the subtle difference, in the case of a string field, between a field being missing, and the field being present but set to the empty string. e.g.

```
<f fid='1'></f>
```

means that the field with fid 1 is set to the empty string; whereas if the field is simply missing within the xml data, then the value of that field is undefined. They are not the same.

When a field is undefined, a reader may choose an appropriate default for the field (which in many cases would actually be the empty string).

I.11 Encryption

The following attribute may be used in any <field> tag within the <definition> section to indicate that the data for that field will be presented in encrypted format within the file.

```
encryption='…'
```

where the value of the attribute specifies the encryption method used.

At present, one value is defined for this attribute:

```
encryption='none'
```

and this simply states that no encryption is used. Any value that starts with the letter '_' is an encryption format private to the program authoring the OPSX file containing it.

Values other than ‘none’ but excluding private values starting ‘_’ are reserved until further defined by a later revision of this standard.

I.12 Personal Names and Addresses

The standard supports 2 different methods of representing personal names and addresses, *split* field and *combined* fields. Readers must support both forms. Writers may support either or both forms.

I.12.A Personal Names

A personal name in split form is given as three fields: title, forename (first name) and surname (last name). E.g.

```
<f fid='202'>Mr</f>
<f fid='203'>John</f>
<f fid='204'>Smith</f>
```

A personal name in combined form is simply a single field containing the name. e.g.

```
<f fid='201'>Mr John Smith</f>
```

I.12.B Addresses

An address in split form is given as various fields: Street1, Street2, Street3, City, State/Region, Country, Postcode/Zipcode. E.g.

```
<f fid='220'>The Meadows</f>
<f fid='221'>17, Main Street</f>
<f fid='224'>Mytown</f>
<f fid='225'>Anystate</f>
<f fid='211'>UNITED STATES</f>
<f fid='226'>90870</f>
```

An address in combined form is simply a single string field containing the address. e.g.

```
<f fid='217'>The Meadows&#013;17, Main
Street&#013;Mytown&#013;Anystate&#013;UNITED STATES&#013;90870</f>
```

I.12.C Mandatory Requirement

Writers may output personal names in split format, or combined format, or both formats. Likewise writers may output addresses in split format, or combined format, or both formats. Generation of split format fields for personal names does not imply that addresses will also be output in split format and vice versa – e.g. personal names may be output in split format but addresses might be output in combined format.

Readers must handle both split and combined forms of these fields. This is necessary for interoperability because writers may generate either form.

I.12.D Personal Names in Animal Table

Fields containing personal names are used in the animal table e.g. for the name(s) of breeder(s). There are 4 fields defined for each context in which a personal name can appear in the animal table: 1 field for the combined form of the name, and 3 further fields for the split form.

E.g. if “Mr John Smith” is a breeder of a dog called Bonzo, this could be represented within that dog’s record as the split form:

```
<record>
```

```
<f fid='110'>Champion</f>
<f fid='101'>Bonzo</f>
<f fid='199'>Mr</f>
<f fid='218'>John</f>
<f fid='219'>Smith</f>
...
```

OR the combined form:

```
<record>
<f fid='110'>Champion</f>
<f fid='101'>Bonzo</f>
<f fid='132'>Mr John Smith</f>
...
```

OR both forms:

```
<record>
<f fid='110'>Champion</f>
<f fid='101'>Bonzo</f>
<f fid='132'>Mr John Smith</f>
<f fid='199'>Mr</f>
<f fid='218'>John</f>
<f fid='219'>Smith</f>
...
```

I.13 Field Grouping

Fields may be grouped together within a record using the <g> tag. This can be used to convey linkage between sets of fields.

E.g. if a cat has two registration numbers together with the names of the registering bodies, each pair of number/body fields should be grouped in order to tell the importing program which body field is associated with which number field. E.g.:

```
<record>
...
  <g>
    <f fid='114'>876876</f>
    <f fid='191'>CFA</f>
  </g>
  <g>
    <f fid='114'>64654</f>
    <f fid='191'>GCCF</f>
  </g>
...
```

II Standard Tables and Fields

This section of the specification defines standard TABLEID's and the standard FIELDID's within those tables.

Some fields are supported by all tables. These are the 'user defined fields' as present in some products.

NOTE: The field ID numbering was changed when OPSVERSION was incremented to 2. This was because the ranges used in version 1 left insufficient ranges of unallocated ids for future expansion. Ranges are currently reserved as follows:
1-199: reserved for fields common to all tables
200-499: reserved for fields in the contact table
500-999: reserved for fields in the animal table

II.1 Fields Common to All Tables

Field	Type	Purpose	Multi	Required	Notes

ID			Field Values OK?	?	
1 through 10	String	User defined string fields, #1 through #10	Yes	No	
11 through 20	Date	User defined date fields, #1 through #10	Yes	No	
21 through 30	Bool	User defined boolean fields, #1 through #10	Yes	No	
31 through 40	Currency	User defined currency fields, #1 through #10	Yes	No	
41 through 50	Int	User defined integer fields, #1 through #10	Yes	No	
101	String	InputSource	No	No	The source of this record – a general purpose text field to indicate the source of this record. E.g. 'KC Stud book, March 2001' etc
102	Date	InputDate	No	No	A date associated with the InputSource field
103	String	UpdateSource	No	No	As InputSource, but this is for a record update.
104	Date	UpdateDate	No	No	A date associated with the UpdateSource field

II.2 Animal Table

TABLEID='1'

There are a lot fields defined here, as this attempts to be the superset of all fields in the programs of OPS members at the time of writing who have contributed their field lists. Where different products have the same data, but it is stored in different ways, the field definitions have been generalized as far as possible.

Field ID	Type	Purpose	Multi Field Values OK?	Required?	Notes
500	String	Name	No	Yes	The official name of the animal, excluding any prefix or suffix titles.
501	String	Pet name	No	No	The 'call' or pet name.
502	Bool	Sex	No	No	Boolean false (0) means the female sex. Boolean true (1) means the male sex. If the sex of the animal is not known, the field is not included within a record. (i.e it is indeterminate).
503	Bool	Sex Proven	No	No	Whether or not the sex is proven. Useful with birds.
504	String	Sexing Method	No	No	Useful with birds.
505	Bool	Status	No	No	Whether or not the animal is neutered/spayed/altered (false, i.e. 0) or not neutered/spayed/altered ('entire') (true, i.e. 1).
506	String	Sire	No	No	The name of the Sire of this animal. If the sire is not known, the field is not included within a record. (i.e. it is

					indeterminate). This name must not include any prefix or suffix titles of the sire – i.e. it is the official name of the sire. If there is a separate record for the sire in the same OPSX file, it will be identical to the value of the name field (id 500) in the sire record.
507	String	Dam	No	No	The name of the Dam of this animal. If the dam is not known, the field is not included within a record. (i.e. it is indeterminate). This name must not include any prefix or suffix titles of the dam – i.e. it is the official name of the dam. If there is a separate record for the dam in the same OPSX file, it will be identical to the value of the name field (id 500) in the dam record.
508	String	Mate	No	No	The name of the usual mate for this animal. (Relevant to birds). This name must not include any prefix or suffix titles of the mate – i.e. it is the official name of the mate. If there is a separate record for the mate in the same OPSX file, it will be identical to the value of the name field (id 500) in the mate record.

509	Date	Date of birth	No	No	It is allowed for the date in the month (DD) to be zero. E.g.:
					`<f fid='509'>20011000</f>`
					means 'born in October 2001'.
					
					Likewise it is allowed for both DD and MM to be zero. E.g
					`<f fid='509'>20010000</f>`
					means 'born in 2001'

520	String	Pre-title	Yes	No	A pre-title for the animal. A 'pre-title' is one that is normally displayed in front of the animal's name.
					
					e.g.
					`<f fid='520'>Champion</f>`
					
					If a writer stores both 'long' and 'short' forms of the pre-title for an animal, e.g. Champion and CH, then both forms should be coded with id 520, but the short form should also have attribute short='1'. E.g.:
					
					`<f fid='520'>Champion</f>`
					`<f fid='520' short='1'>CH</f>`
					
					If a writer stores only one form of the field, as will be true for most writers, it should be coded without the 'short' attribute.
					
					Multiple titles may be given by repeating the field multiple times.

522	String	Post-title	Yes	No	A string giving a post-title for the animal. Operates just like field id 520 but for titles that are normally placed AFTER the animal's name e.g. in pedigrees. If a writer stores both 'long' and 'short' forms

					of the after title for an animal, e.g. Distinguished Merit and DM, then both forms should be coded with id 522, but the short form should also have attribute short='1'. E.g.: <f fid='522'>Distinguished Merit</f> <f fid='522' short='1'>DM</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the 'short' attribute.
524	String	Pre-nontitle.	Yes	No	This defines 'prefixes' that can come in front of an animal's name but which are neither pre-titles nor part of the animal's official name.
525	String	Post-nontitle.	Yes	No	This defines 'suffixes' that can come after an animal's name but which are not actually post-titles nor part of the animal's official name. E.g: <f fid='514'>US Import</f>
530	String	Registration Number	Yes	No	Registration number. Preferably should not include the name of the registering body – the body should go in field 531 if possible. Fields 531-533 are defined for any associated registering body name, date of registration, and country of registration. When any of these fields are present they should be grouped with the 530 field. E.g. <g> <f fid='530'>CS987987</f> <f fid='531'>GCCF</f> <f fid='532'>20020830</f> <f fid='533'>UK</f> </g> Grouping is essential when an animal has multiple registration numbers AND there are 531-533 fields present.
531	String	Registering Body	Yes	No	Name or abbreviation of the registering body (registry) associated with the registration number given by field id 530. Normally only present when field id 530 is also present, and typically will be grouped with field 530 and, if present, 532/533. See notes for field 530.
532	Date	Registration Date	Yes	No	Date of registration associated with the registration number given by field id 530. Normally only present when field id 530 is also present, and typically will be grouped with field 530 and, if present, 531/533. See notes for field 530.
533	String	Country of Registration	Yes	No	Country of registration associated with the registration number given by field id 530. Normally only present when field id 530 is also present, and typically will be grouped with field 530 and, if present, 531/532. See notes for field 530.

540	String	Stud Book Number	Yes	No	Stud Book number. Preferably should not include the name of the publishing body – the body should go in field 541 if possible. Fields 541-542 are defined for any associated publishing body name and date of publication. When either of these fields are present they should be grouped with the 540 field. E.g. <g> <f fid='540'>SB02</f> <f fid='541'>KC</f> <f fid='542'>20020830</f> </g> Grouping is essential when an animal has multiple stud book numbers AND there are 541-542 fields present.
541	String	Stud Book Publisher	Yes	No	Name or abbreviation of the publisher associated with the stud book number given by field id 540. Normally only present when field id 540 is also present, and typically will be grouped with field 540 and, if present, 542. See notes for field 540.
542	Date	Stud Book Publication Date	Yes	No	Date of publication of the stud book number given by field id 540. Normally only present when field id 540 is also present, and typically will be grouped with field 540 and, if present, 541. See notes for field 540.
543	String	Litter Reg No.	No	No	The litter registration number of the litter of which this animal was part.
544	String	Microchip Number	Yes	No	
545	String	Tattoo Number	Yes	No	
546	String	Cage	No	No	Cage or hutch number.
560	Date	Date of Death	No	No	Comments from DOB apply here too.
561	String	Cause of Death	No	No	
562	String	Weight	No	No	
563	String	Height	No	No	Also known as 'Size' in some products.
564	String	Length	No	No	
565	String	CERF number	No	No	
566	String	OFA number	No	No	
567	String	Hip Score	No	No	This is a free format text field. No syntax is defined for hip scores.
568	String	Elbow Score	No	No	This is a free format text field.
569	String	Genetic Defects	Yes	No	No format defined for this field. Multiple defects may be given as multiline text (separated by carriage return character entities i.e
), or may be given as multiple instances of the field.
570	String	Temperament	No	No	
571	String	Blood type	No	No	
572	String	Eye test result	No	No	Result of the most recent eye test.

573	Date	Eye test date	No	No	Date of most recent eye test.
574	String	Health notes	No	No	General purpose health memo field. Typically multiline.
575 to 578	String	Usual Vet	No	No	The name of the usual vet of this animal. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. Field IDs: 575 – combined form name 576 – split form, title 577 – split form, forename(s) 578 – split form, surname
600	String	Country of Import	No	No	
601	Date	Date of Import	No	No	
602	String	Import Number	No	No	
603	String	Country of Export	No	No	
604	Date	Date of Export	No	No	
605	String	Export Number	No	No	
606 to 609	String	Sold To	No	No	Name of person this animal was sold to. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. Field IDs: 606 – combined form name 607 – split form, title 608 – split form, forename(s) 609 – split form, surname
610	Date	Sold On	No	No	Date on which animal was sold
611	currency	Sold For	No	No	Amount animal was sold for.
612 to 615	String	Bought From	No	No	Name of person from whom this animal was bought. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. Field IDs: 612 – combined form name 613 – split form, title 614 – split form, forename(s) 615 – split form, surname
616	Date	Bought On	No	No	Date animal was bought
617	currency	Bought For	No	No	Amount animal was bought for
630 to	String	Breeder	Yes	No	The name of the breeder.

633					Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. When multiple breeders are present AND split form fields are used, grouping must also be used on each tuple of fields 631-633. Field IDs: 630 – combined form name 631 – split form, title 632 – split form, forename(s) 633 – split form, surname
634 to 637	String	Owner	Yes	No	The name of the owner. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. When multiple owners are present AND split form fields are used, grouping must also be used on each tuple of fields 635-637. Field IDs: 634 – combined form name 635 – split form, title 636 – split form, forename(s) 637 – split form, surname
638 to 641	String	Surveyor	No	No	Name of the surveyor. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.D for further details. Field IDs: 638 – combined form name 639 – split form, title 640 – split form, forename(s) 641 – split form, surname

700	String	Coat Colour	No	No	Coat colour. That part of the full breed description that truly relates to coat colour. E.g. with a Chocolate Tabby Point Siamese, the coat colour would be 'Chocolate'.
					
					If a writer stores both ‘long’ and ‘short’ forms of the coat colour for an animal, both forms should be coded with id 700, but the short form should also have attribute short=’1’. E.g.:
					
					`<f fid=’700’>Chocolate</f>`
					`<f fid=’700’ short=’1’>Choc</f>`
					
					If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute.
					
					A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.

702	String	Coat Pattern	No	No	Coat pattern. That part of the full breed description that truly relates to coat pattern. E.g. with a Chocolate Tabby Point Siamese, the coat pattern would be ' Tabby Point' or perhaps just 'Tabby'.
					
					If a writer stores both 'long' and 'short' forms of the coat pattern for an animal, both forms should be coded with id 702, but the short form should also have attribute short='1'. E.g.:
					
					`<f fid='702'>Tabby</f>`
					`<f fid='702' short='1'>Tby</f>`
					
					If a writer stores only one form of the field, as will be true for most writers, it should be coded without the 'short' attribute.
					
					A reader that does not support both short and long forms of the field should simply ignore the 'short' attribute.

704	String	Eye Colour	No	No	Eye colour.
					
					If a writer stores both 'long' and 'short' forms of the eye colour for an animal, both forms should be coded with id 704, but the short form should also have attribute short='1'. E.g.:
					
					`<f fid='704'>Odd Eyed</f>`
					`<f fid='704' short='1'>Odd</f>`
					
					If a writer stores only one form of the field, as will be true for most writers, it should be coded without the 'short' attribute.
					
					A reader that does not support both short and long forms of the field should simply ignore the 'short' attribute.

706	String	Variety	No	No	Variety. Equivalent to the full breed description, less the breed. E.g. with a Chocolate Tabby Point Siamese, the variety would be ' Chocolate Tabby Point '.
					
					If a writer stores both 'long' and 'short' forms of the variety for an animal, both forms should be coded with id 706, but the short form should also have attribute short='1'. E.g.:
					
					`<f fid='706'>Chocolate Tabby Point</f>`
					`<f fid='706' short='1'>Choc Tby Pt</f>`
					
					If a writer stores only one form of the field, as will be true for most writers, it should be coded without the 'short' attribute.
					
					A reader that does not support both short and long forms of the field should simply ignore the 'short' attribute.

708	String	Breed	No	No	This is the overall breed of the animal.

					E.g. with a Chocolate Tabby Point Siamese, the breed would be 'Siamese'. If a writer stores both ‘long’ and ‘short’ forms of the overall breed for an animal, both forms should be coded with id 708, but the short form should also have attribute short=’1’. E.g.: <f fid=’708’>Siamese</f> <f fid=’708’ short=’1’>SIA</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
710	String	Full Breed	No	No	This is the complete description of the type of animal. E.g. with a Chocolate Tabby Point Siamese, this is literally 'Chocolate Tabby Point Siamese'. If a writer stores both ‘long’ and ‘short’ forms of the full type for an animal, both forms should be coded with id 710, but the short form should also have attribute short=’1’. E.g.: <f fid=’710’>Chocolate Tabby Point Siamese</f> <f fid=’710’ short=’1’>Choc Tby Pt SIA</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
712	String	Group	No	No	This is the breed group. E.g with dogs, this might be 'Toy', 'Utility' etc. E.g. with cats, this might be 'Oriental', 'Shorthair' etc. If a writer stores both ‘long’ and ‘short’ forms of the breed group for an animal, both forms should be coded with id 712, but the short form should also have attribute short=’1’. E.g.: <f fid=’712’>Pastoral</f> <f fid=’712’ short=’1’>Past</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
714	String	Ear Type	No	No	Ear type e.g. Folded, Lop Eared etc.

					If a writer stores both ‘long’ and ‘short’ forms of the ear type for an animal, both forms should be coded with id 714, but the short form should also have attribute short=’1’. E.g.: <f fid=’714’>Lop Eared</f> <f fid=’714’ short=’1’>Log</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
716	String	Tail Type	No	No	Tail type e.g. Manx, Stumpie etc If a writer stores both ‘long’ and ‘short’ forms of the tail type for an animal, both forms should be coded with id 716, but the short form should also have attribute short=’1’. E.g.: <f fid=’716’>Rumpy Riser</f> <f fid=’716’ short=’1’>RR</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
718	String	Fur Type	No	No	Fur type e.g. ‘Long haired. If a writer stores both ‘long’ and ‘short’ forms of the fur type for an animal, both forms should be coded with id 718, but the short form should also have attribute short=’1’. E.g.: <f fid=’718’>Long Haired</f> <f fid=’718’ short=’1’>LH</f> If a writer stores only one form of the field, as will be true for most writers, it should be coded without the ‘short’ attribute. A reader that does not support both short and long forms of the field should simply ignore the ‘short’ attribute.
720	String	Genus	No	No	
721	String	Species	No	No	
722	String	Sub Species	No	No	
723	String	Breed Number/Code	Yes	No	Breed number or code. A number or code that identifies the breed/type. Typically different registries use different coding schemes. E.g. with cats there are GCC breed numbers, CFA breed numbers (different to GCCF numbers), EMS codes (as used by FIFe

					breeders), etc.
					
					Writers should use the 723 field for the number/code itself. Optionally, the name of the associated registry may be given in a 724 field.
					
					Multiple breed numbers/codes are allowed. When multiple numbers/codes are present AND there are 724 fields present, grouping should be used to avoid ambiguity.
					
					E.g. a Chocolate Burmese cat could be assigned GCCF breed number '27b', EMS code 'BUR b'. or CFA breed number '1402'. A writer that emitted all three forms could do it like this:
					
					`<g>`
					`  <f fid='723'>27b</f>`
					`  <f fid='724'>GCCF</f>`
					`</g>`
					`<g>`
					`  <f fid='723'>BUR b</f>`
					`  <f fid='724'>FIFE</f>`
					`</g>`
					`<g>`
					`  <f fid='723'>1402</f>`
					`  <f fid='724'>CFA</f>`
					`</g>`

724	String	Breed Number/Code Issuing Body	Yes	No	See field 723.
725	String	Faults	No	No	i.e faults when compared to the breed standard
726	String	TICA Category	No	No	Cat specific: this is the TICA category
727	String	TICA Division	No	No	Cat specific: this is the TICA division
728	Bool	Non-Standard Breed?	No	No	Is this a non-standard breed?
750	String	Genotype	No	No	
751	String	Coat Pattern Genotype	No	No	
752	String	Coat Colour Genotype	No	No	
753	String	Eye Colour Genotype	No	No	
754	String	Defect Genotype	No	No	
800	String	Gestation Period	No	No	A number of days.
801	String	Points	No	No	
802	String OR image/jpeg OR image/bmp OR image/gif OR application/x-msmeta	Picture	Yes	No	A picture(image). This can either be given as a filename, in which case the field is of type 'string'. OR It can be given as the image data itself, in which case the field type is one of the supported image formats, and the 'type' attribute must be given.

	file OR application/x-msenhancedmetafile				
803	String	Memo	No	No	Free format text field. Generally multi-line.
804	String	Comments	No	No	Another free format text field. Generally multi-line.
805	String	Legs	No	No	Relevant to rabbits, cavies etc. This is the #legs won at shows
806	String	GC number	No	No	Relevant to rabbits etc. This is a number assigned to the animal upon reaching Grand Champion status.
807	String	Win	Yes	No	Again relevant to rabbits etc. A 'win' is simply a bit of text describing a show win. There may be multiple 'wins'.

II.3 Contact Table

TABLEID='2'

Field ID	**Type**	**Purpose**	**Multiple Field Values Allowed?**	**Required?**	**Notes**
201 to 204	String	Name	No	Yes	Name of the person. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.A and I.12.C for further details. Field IDs: 201 – combined form name 202 – split form, title 203 – split form, forename(s) 204 – split form, surname
205 to 212	String	Address	No	No	The postal address. Writers may encode this in split form, or combined form, or both split and combined forms. Reader must support import in either split or combined form. See section I.12.B and I.12.C for further details. Field IDs: 205 – combined form address 206 – split form, Street1 207 – split form, Street2 208 – split form, Street3 209 – split form, City 210 – split form, State/Region 211 – split form, Country 212 – split form, Postcode/Zip code
213	String	Home Phone	Yes	No	
214	String	Work Phone	Yes	No	

215	String	Cell (Mobile) Phone	Yes	No	
216	String	Fax number	Yes	No	
217	String	Email address	Yes	No	
218	String	URL (website)	Yes	No	
219	Date	Date of Birth	No	No	
220	String	Club	Yes	No	Name of club this contact is a member of. Multiple clubs allowed by repeating the field.
221	String	Kennel number	Yes	No	Kennel/cattery number (if assigned by governing body).
222	String	Kennel name	Yes	No	Names of kennels (dogs), prefixes/catteries (cats), etc that this contact is associated with. Multiple names allowed, by repeating the field.
223	String	Judge of	Yes	No	Name of a breed this person is a recognized judge of.
224	String	Show Manager for	Yes	No	Name of show this person is usually a show manager of. Multiple names allowed, by repeating the field.
225	String	Notes	No	No	Free format notes field.

II.4 Other Tables

To be defined.